SHIP PRODUCTION COMMITTEE
FACILITIES AND ENVIRONMENTAL EFFECTS
SURFACE PREPARATION AND COATINGS
DESIGN/PRODUCTION INTEGRATION
HUMAN RESOURCE INNOVATION
MARINE INDUSTRY STANDARDS
WELDING
INDUSTRIAL ENGINEERING
EDUCATION AND TRAINING

April 1997
NSRP 0532

THE NATIONAL SHIPBUILDING RESEARCH PROGRAM

1997 Ship Production Symposium

Paper No. 28: A Prototype Object-Oriented CAD System for Shipbuilding

U.S. DEPARTMENT OF THE NAVY
CARDEROCK DIVISION,
NAVAL SURFACE WARFARE CENTER

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
APR 1997	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
The National Shipbuilding Research Program 1997 Ship Production Symposium, Paper No. 28: A Prototype Object-Oriented CAD System for Shipbuilding	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Surface Warfare Center CD Code 2230-Design Integration Tower Bldg 192, Room 128 9500 MacArthur Blvd Bethesda, MD 20817-5700**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**12**	

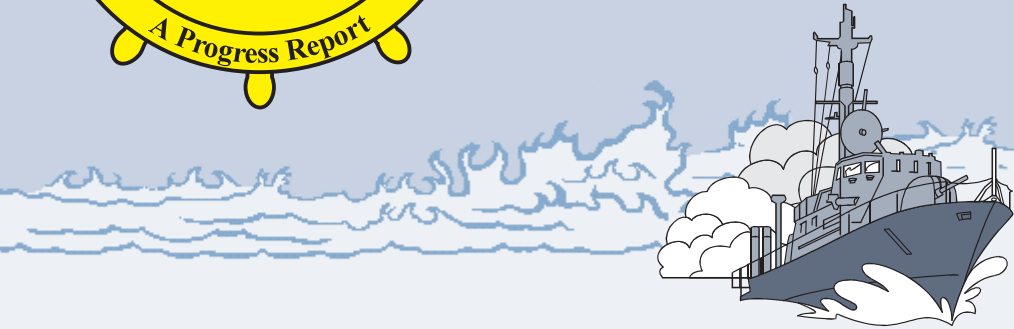
DISCLAIMER

These reports were prepared as an account of government-sponsored work. Neither the United States, nor the United States Navy, nor any person acting on behalf of the United States Navy (A) makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness or usefulness of the information contained in this report/ manual, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or (B) assumes any liabilities with respect to the use of or for damages resulting from the use of any information, apparatus, method, or process disclosed in the report. As used in the above, “Persons acting on behalf of the United States Navy” includes any employee, contractor, or subcontractor to the contractor of the United States Navy to the extent that such employee, contractor, or subcontractor to the contractor prepares, handles, or distributes, or provides access to any information pursuant to his employment or contract or subcontract to the contractor with the United States Navy. ANY POSSIBLE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR PURPOSE ARE SPECIFICALLY DISCLAIMED.

International Competitiveness By the Year 2000

SPC

A Progress Report



THE SOCIETY OF NAVAL ARCHITECTS AND MARINE ENGINEERS
1997 Ship Production Symposium
April 21-23, 1997
New Orleans Hilton Hotel
New Orleans, Louisiana

THE SOCIETY OF NAVAL ARCHITECTS AND MARINE ENGINEERS
601 Pavonia Avenue, Jersey City, NJ 07306
Tel. (201) 798-4800 Fax. (201) 798-4975
Paper presented at the 1997 Ship Production Symposium, April 21-23, 1997
New Orleans Hilton Hotel, New Orleans, Louisiana

A Prototype Object-Oriented CAD System For Shipbuilding

Norman L. Whitley (V), University of New Orleans,

ABSTRACT

This paper reports on the on-going development of an object-oriented CAD system at the Advanced Computer Laboratory for Shipbuilding at the University of New Orleans. It describes a) the reasons for object-oriented (yard-specific) development, b) the computer-aided software development environment, c) the developing class structure of the ship structures design application, and d) the planned developments within the CAD system and integration of packages to support visualization, planning and enterprise management and electronic data interchange.

NOMENCLATURE

AP - Application protocol
CAD - Computer-aided design
CE - Concurrent engineering
EDI - Electronic data interchange
IPPD - Integrated product & process development
ISO - International Standards Organization
NSRP - National Shipbuilding Research Program
ODMG - Object Data Management Group
OLE - Object linking & embedding
OODBM - Object-oriented data base manager
OOP - Object-oriented programming
OOT - Object-oriented technology
PC - Personal computer
SBD - Simulation-based design
STEP - STandard for the Exchange of Product model data
2D - Two-dimensional
3D - Three-dimensional

INTRODUCTION

The work described here is concerned mainly with the use of object technology in software application development. This new technology is changing how computer software is written and maintained. It is changing the paradigms of software development while radically shortening the time scales of that development. The specific project that will be discussed here is that of building a customized computer-aided design system from libraries of geometry, topology, and graphical user interface components. These libraries include entities and algorithms. The goals of this project were to use object technology to create a working prototype of a ship design system, assess its advantages and weaknesses as compared to commercially available systems, and assess the feasibility (or economic justifiability) of in-house development within American shipyards.

The status of this work will be discussed along with sufficient background in object programming and technology, as well as current business trends in information resources, design and manufacturing. There will be some discussion of future efforts in the use of a centralized database of which the product model is an essential element.

BACKGROUND

Generic Computer-Aided Design

The past fifteen years have seen the appearance of computer-aided design (CAD) systems within the design, manufacture, and engineering components of companies worldwide. This has been mostly due to the plummeting price of computing power and the availability of interface-driven operating systems and powerful application packages. The ability to prescribe, describe, and analyze products using computer programs as a primary tool has allowed for a pronounced change in the way that products are conceived.

The question of how to chose a CAD system that will lead to greater success for a company is a difficult one to answer. Cost-benefit analyses are not totally successful in that they do not reflect the culture of a company, and are often based on processes that currently exist and will or should not in the future. These analyses tend to be viewed as pre-arranged - the figures were made to justify the desired outcome. An excellent overview of the difficulty in choosing a CAD system can be found in Marks and Riley (1). This book also offers a superb scheme by which a rational choice could be made.

The phrase “CAD” has taken on numerous meanings due to the vast differences in the scope and power of commercially available packages. CAD can mean as little as creating 2D line drawings (drafting). It can mean as much as creating 3D solid models whether through constructive solid geometry or through 3D boundary representations with topology.

Low end CAD packages are in some ways not very powerful, but still may be viewed as being complex to the uninitiated. They run on personal computers (PC’s) running various versions of the Microsoft Windows, Macintosh, or DOS operating systems. Vendors in this area, which are numerous, include AutoDesk, Cadkey, and Ashlar.

These packages also allow for connection to various external software programs that may perform analysis, 3D visualization, or management functions. This connection may be straightforward, accomplished by operating system function, such as MicroSoft Windows Object Linking and Embedding (OLE), or through saved file structure. Often it is the case that the connection is cumbersome, requiring file manipulation that is difficult to automate.

Regardless, these higher level functions are not the key to the succes of these low-end packages. Historically a company’s manufacturing processes have been based on 2D line drawings. The

craftsmen that build the product have extensive experience in interpreting this kind of design output. In this sense, 2D line drawings do reflect manufacturing process, they are the seminal element from which manual manufacturing proceeded. It is for this reason alone that low-end CAD packages have been such a success.

High end packages are very powerful and indeed very complex. They run on very high-end PC's or on workstations running some version of the UNIX operating system. They have numerous operations and features in an attempt to surround all possible design algorithms and they include copious optional modules for analysis, manufacturing, and/or management, etc. Vendors in this area include Parametric Technologies, IBM, SDRC, ComputerVision, and Intergraph.

Even though these systems offer enormous power they are not the automatic preference, even for companies that need more than 2D line drawings as design output (for example they may need toolpaths for numerically controlled machines or robots). Some of the problems with these packages are:

- their cost - initial cost, the cost of lost productivity while personnel learn the new system and adapt to it, and the on-going cost of relatively sophisticated systems and computers that require maintenance
- their difficulty to master - their powerful structure leads to complicated interfaces with abundant selections and complex command sequences
- their lack of open communications - even though they have abundant modules for support they don't directly communicate with the company's well honed materials management system
- their overhead of features - they have a large number of options for doing certain tasks, many more than a company will need or use
- their inability to reflect a company's design practices and manufacturing processes.

For these reasons, some businesses have taken off-the-shelf packages and have over time tailored them to the processes and practices of their yard. This is usually a difficult and expensive task but results in a highly effective CAD package. This is essentially how Boeing Aircraft has developed its world renown CAD system. Although it is CATIA, the developers of CATIA, Dassault and IBM, worked extensively with Boeing to provide the functionality that Boeing required (2).

The Information Age. As companies move to cut manufacturing costs through automation and process improvement, it is crucial that information from design be changed to support the new manufacturing methods. To be competitive companies need to be responsive and capitalize on what they do well. The right kind of information at the right place is necessary for optimal operation. Information, in fact, and its management is now the focal point of corporate competitiveness. Creating information and storing it in a central database that is then shared, modified and utilized by all internal units is seen as essential to being competitive.

At the heart of this database is the three dimensional product model (3) which consists of the 3D geometry and topology of the product and its parts, its material properties, its manufacturing processes, its relationships to all other products, maintenance requirements, etc. The database also contains marketing information that may include 3D visualizations, or virtual reality presentations, purchasing information, financial information, etc.

This view of centralized information as the chief company asset is developing in conjunction with the philosophy of integrated product and process development (IPPD) or concurrent engineering (CE). On a philosophical level IPPD is a frontal attack on the design of a product. It is all business units acting simultaneously in combination with the customer to create a design. On a functional level, IPPD cannot succeed as it is intended unless there is high level integration of methodologies and tools, seamless communication between working groups, and a shared database that defines the product. The core of the functionality of IPPD is computers, networks, and information technology. A deficiency in most current CAD systems is that they are very much design and engineering systems. They are not business systems. They are not a ready part of the new IPPD world.

A new facet of IPPD that is currently emerging in manufacturing is simulation based design (SBD). SBD is the practice of using product design knowledge in simulations and visualizations during the design process. As much as possible, the product is "tested" and "reviewed" using software and computers before manufacturing starts. This means physics-based simulation and virtual reality evaluations of the product's structure. SBD requires the geometry of the product as well as knowledge about its physical properties. The 3D product model is needed for this process.

STEP. Another element that is playing a role in the future of CAD is STEP- STandard for the Exchange of Product model data, a standard of ISO (10303). Within STEP are conventions for basic geometry and topology. On these conventions are built higher level entities that are industry specific and these are collected in application protocols (AP). STEP infers a standard format for exchanging CAD data between different software systems. Much of the world is adopting this standard. It will be the neutral format for exchanging data and yet most CAD systems do not have the STEP definitions as part of their basic elements. STEP translators must be created to take a vendor's format (AutoDesk's DXF for instance) and convert it to this neutral format and vice versa. This is not a trivial task. Many CAD systems store geometry and not topology, or their topology is not robust or consistent with STEP. There is currently a funded effort (4) to create prototypes of these translators.

These issues point to the need for a new generation of CAD systems that are part of the whole business process, that are modular, flexible, extendible, and can be tailored to suit a company's strength. These new systems need to provide data that is available to all business units and can be transmitted easily to business partners and customers. Two recent brief articles by Deitz (5,6) review new CAD systems in this light.

Shipbuilding Computer-Aided Design

The level of use of computer aids in American yards (and their impact) has been well documented. Important recent works include NSRP report 0373 (7), and the papers of Storch, Clark, and Lamb (8) and Ross and Garcia (9). A broad overview of computer aids in all aspects of ship manufacture can be found in Latorre and Zeidner (10). A paper by Storch and Chirilo (11) speaks squarely to effectively using CAD for more than basic design function.

Concurrent engineering is being strongly promoted by the branches of the U. S. armed services and is being embraced by several shipyards. It was the topic of three recent NSRP efforts. They are documented in reports 0435 (12), 0436 (13), and 0454 (14).

There are CAD packages that are specifically for shipbuilding. These off-the-shelf products include Autoship from Autoship Systems, FAST SHIP from Proteus, ISDP from Intergraph, FORAN from Senemar, and Tribon from KCS. Both IBM's CATIA and Parametric Technology's Pro/Engineer have recently included ship design packages in their optional modules. These packages like the

generic ones differ greatly in scope and power. Each has strengths and weaknesses. These may not be a perfect fit for any yard but could be profitable solutions in many yards.

The functionality of world-class CAD systems is being reviewed and characterized in an ongoing project funded by the NSRP (15). This project is at a midway point but its interim report supports this idea: world-class does not have to mean cutting-edge technology, but it does mean highly tailored systems that capture and enable what your company does and supports it as much as possible.

As an example that is somewhat different from Boeing's effort with CATIA there is the Danish yard at Odense and the Hitachi Zosen yard in Japan which have developed HICADEC, one of the most successful computer aids in shipbuilding. This development has almost totally been done in-house over many years, but HICADEC has become a powerful tool for these yards which are considered to be among the most competitive and productive in the world.

A significant effort in this area of customization is that of Newport News Shipyards. This shipyard participated in a DARPA funded project for the development of simulation-based design (Lockheed/Martin was the lead contractor). As part of that Newport News has created a smart product modeling system for shipbuilding. This system's architecture is based on several commercial-off-the-shelf products. Entities are created in the 3D CAD environment, placed in a database, and managed by an object-oriented database manager. This allows for those entities (objects) to possess attributes of almost any nature. The information can be queried at any time by the database manager. New information can be attached to an object at any time. Thus, a smart 3D product model exists.

It is the success of these in-house developments that stimulated this research. The lessons that could be drawn were:

- The more a yard could tailor the CAD system to their processes and practices the more valuable it was. For CAD systems to be of the most value they had to be flexible, modular and open.
- The users had to be able to determine their characteristics and functionality. The users had to be able to institute new algorithms that are useful to them alone. They had to be able to remove all functionality that is of no use to them.
- They had to be able to create any standard entity that is necessary for their design or manufacturing, even it is only a standard for them. They had to be able to use terminology that is the practice of their yard.

The success of these in-house developments is so clear one may ask if something similar is the answer for every shipyard. If given the opportunity by software vendors, most yards could eventually tailor commercial products into something extraordinary for their own use. But these developments may take many years, and that is time that American shipyards do not have. They must become competitive on the world market in the immediate future or many will not survive.

One may also ask if something like the Newport News system is the answer. With a product like that one there are dangers in that the future is not totally controlled by the yard:

- The component commercial-off-the-shelf products will evolve and may not remain the component that they need.
- It may not be possible to include newly identified functionality requirements in those core products at a later time.
- It may be that the communication between these products will not always remain clear and seamless.

The questions that motivated this research are: Is it feasible for a yard to build a self-contained state-of-the-art CAD system (a smart 3D product modeling system) - one whose function and input/output can be integrated into all business processes - from scratch? If feasible, what expertise does it require? How many people would it require? What is the time-frame of such a development?

OBJECT-ORIENTED PROGRAMMING LANGUAGES AND PARADIGMS

Object-oriented technology (OOT) is an extension of the paradigms upon which object-oriented programming (OOP) was built. OOP languages are the most current fad in the computer science community. These languages are relatively new (the oldest is about 30 years old) but have really stormed to the front in the world of application development within the last 5 years or so. They are emerging as the unanimous choice for building applications that are centered around the creation, management, and sharing of information.

Computer languages that are most familiar to people like Fortran, C, and Basic are of the oldest type and are called procedural languages. They are used to create procedures for doing calculations or manipulating data, etc. The popular languages just prior to OOP were structured procedural languages. The motivation behind these languages (it was more a style than a new language) was verification of code. Large pieces of code were difficult to verify if the code lacked a formal structure.

OOP languages have very formal structures and that is one of their strengths This structure is based on several definitions and paradigms, some of which will be presented below. OOP languages obviously execute procedures. With OOP, it is how procedures are packaged that is significant.

Detailed information about object-oriented program-ming and technology can be found in numerous books. Among them are those by Meyer (15), Kemper (16) and Burleson (17). A less technical overview can be found in the book by Taylor (18), and a less optimistic view is provided by Webster (19).

OOP Structure

In OOP language programs data and the procedures that operate on that data are packaged together in *objects*, pieces of code that are self-contained in a somewhat similar way that sub-routines in procedural languages are self-contained. Procedures are never written such that they are unattached to data. A *class* is a template for a set of similar objects. A class is a package that contains all of the procedures (called *methods*) and variables for every member of the set. Creating a class avoids needless redundancy of code.

What follows is a short description of four important concepts for object-oriented languages. These four traits embody the power of these languages to improve the structure and design of programs.

Abstraction. The ability to create classes that represent a certain set of data as a new data type is called abstraction. In most procedural languages there are pre-defined data types: real, integer, character, boolean, etc. It is not possible to create a new type of data. In OOP every class can be considered to be an abstract data type. A class represents a whole new data structure that has well defined behaviors and characteristics.

Encapsulation. The feature of packaging together corresponding variables and methods within an object is called encapsulation. It is important because it allows for the details of procedures to be hidden from outside the object. Methods are never passed to objects, only messages. The message asks for some method to execute but the details of the method are not known to the sender of the message. This allows for simple interaction between objects and therefore for easy modification of the methods without

wholesale changes of the code.

Inheritance. The acquisition of methods and variables by a class simply by its position in a hierarchy is called inheritance. All classes are placed in a hierarchy (in some OOP languages multiple hierarchies are allowed). Classes have descendant (or sub) classes that inherit their methods and variables. They have parent (or super) classes from which they inherit. This is a property that eliminates redundancy and encourages consistency.

A program contains classes for closed polygons, quadrilaterals, rectangles, squares, triangles, and isosceles triangles. In the hierarchy, quadrilaterals and triangles inherit from closed polygons. Rectangles inherit from quadrilaterals and squares inherit from rectangles. Isosceles triangles inherit from triangles. When the message is sent to any member of the class square to provide its area, an appropriate method executes. A "compute area" method could exist in the class square, but one also exists in the class rectangle. The class square could inherit the method of computing area from the class rectangle, which may or may not inherit the method from the class quadrilateral. The same scenario exists for the triangle branch of the hierarchy.

Polymorphism. The ability to hide different responses to a single message behind an object's interface is referred to as polymorphism. In the hierarchy above, if the message of "provide area" is sent to a member of the class square or triangle, they both respond with their areas even though the method used to compute the areas is different. The message sent is simple - "provide area." The response it elicits is the same as seen from outside the object. This feature of OOP allows for simple and consistent interaction between objects.

OOP Languages and Database Managers

There are pure OOP languages and there are hybrid ones. The most important of these would include Simula (the original), SmallTalk, and now Java, which are pure OOP languages. Objective-C and C++ are hybrid OOP languages, both being OOP extensions of the language C. Both of these languages allow for procedural code to exist along with object-oriented code. They were created to take advantage of the power of C at doing some procedural tasks.

The language chosen for this development is C++. C++ can be said to be arcane and has some very challenging features that are not good for beginning programmers, but at this time it is the most commonly employed OOP language. There is no ANSI (or other) standard for C++ at this time, which means that every vendor's C++ compiler has different capabilities.

Object-oriented database managers (OODBM) use the paradigms set forth above. Because they do they offer a powerful way to store complex data structures. Relational databases were designed to store conventional data types: real numbers, character strings, boolean values, etc. When you have created a hierarchy of objects, each of which can be considered to be an abstract data type, relational databases cannot directly store that information. The OODBM can store that information just as it is and can then query it. It does so by storing references between a class and its instances, between objects and other objects. So a composite piece has references to all of its components - all of the variables that are related to it. They could be character strings, real numbers, topological characteristics, geometry, a rasterized drawing, a bill of materials, etc.

The manipulation and communication of objects as described above is standardized by a working group called the Object Data Management Group. Their standard ODMG-93 is generally accepted in this area.

OOT Conclusion

It is because of these traits of object-oriented technology that it is currently the choice for development of complicated software applications. It offers the ability to build applications in a highly modular way with abstract data types of any nature. Data and procedures are always associated with their pertinent objects. This leads to code structure that can be more easily verified to work. Changing code to include new features or to modify existing ones can be done with limited re-writing of existing code. OOT leads to data structures that can be highly heterogeneous and yet very usable.

In closing, the reader is reminded that the word "object" is used in a lot of different contexts concerning computers. One of the most frequent uses is in conjunction with MicroSoft's Object Linking and Embedding (OLE). This technology is very different from what is described here. Not all of the paradigms listed above actually pertain to OLE. OLE is a very rigorous and useful standard but one that only exists in MircoSoft's Windows operating systems.

DEVELOPMENT ENVIRONMENTS

One option for development would be to buy a C++ compiler and start from scratch. That clearly carries a lot of risk. If that were the only option then developing an in-house CAD system would not be justifiable. Fortunately there are toolkits that are available that makes this process possible and warranted. These toolkits include those from ComputerVision Corp. (Pelorus) and Matra Datavision (CAS.CADE - computer-aided software for computer-aided design engineering). The details of these toolkits differ considerably but they have both have the elements needed to create OOP CAD applications. The toolkit or development environment used here is CAS.CADE.

The environment consists of a methodology for creating applications supported by appropriate tools and a set of expandable C++ class libraries. These libraries include classes for modeling, analysis, graphical presentation, graphical user interface implementation using Motif constructs, and data management. There are extensive libraries for creation of geometry and topology, in both 2D and 3D. These two libraries are STEP compliant. The basic entities were created using STEP Part 42 definitions. These libraries support non-manifold topology.

For both of the environments mentioned above finished applications can be deployed on machines running versions of the MicroSoft Windows operating system. They also required a language compiler, either MicroSoft Visual Basic for Pelorus or Visual C++ for CAS.CADE.

The brief description below is meant to impart a notion of possible elements in a robust environment for the development of CAD. The various types of software components (development units) are given these names (see Figure 1.):

- a set of related classes is called a *package*
- a set of data types known to an application database is called a *schema*
- a set of related packages can be formally grouped together into a *toolkit*
- a set of packages, classes, and methods whose services are exported to the front end is called an *interface*.
- a set of interfaces is called an *engine*
- a set of chosen engines make up an *application*

This categorization reflects the modular nature of development.

Pieces are constructed from smaller pieces, and so forth.

In example, an application would include a dialogue engine which implements the ergonomics of the user-interface. It handles all of the user-initiated screen events (whether graphical, button, menu selection, or text) and passes them to the front-end. The front-end is basically the software driver of the engines - it calls scripts that cause messages to be sent to appropriate objects and thus actions are taken. The application engine (there could be more than one) would provide all of the functionality that the user expects in terms of object creation, algorithmic behavior, and data storage.

Referring to Figure 2., the development concepts can be seen. The development is structurally formalized by the use of a definition language. Using this concise language new classes are defined. The definitions are then compiled and the results stored in the data dictionary. At this point the compiler creates an appropriate C++ template and header for all of the methods for all of the defined classes. The user takes these templates and completes them thereby creating his/her desired procedures.

Figure 1.

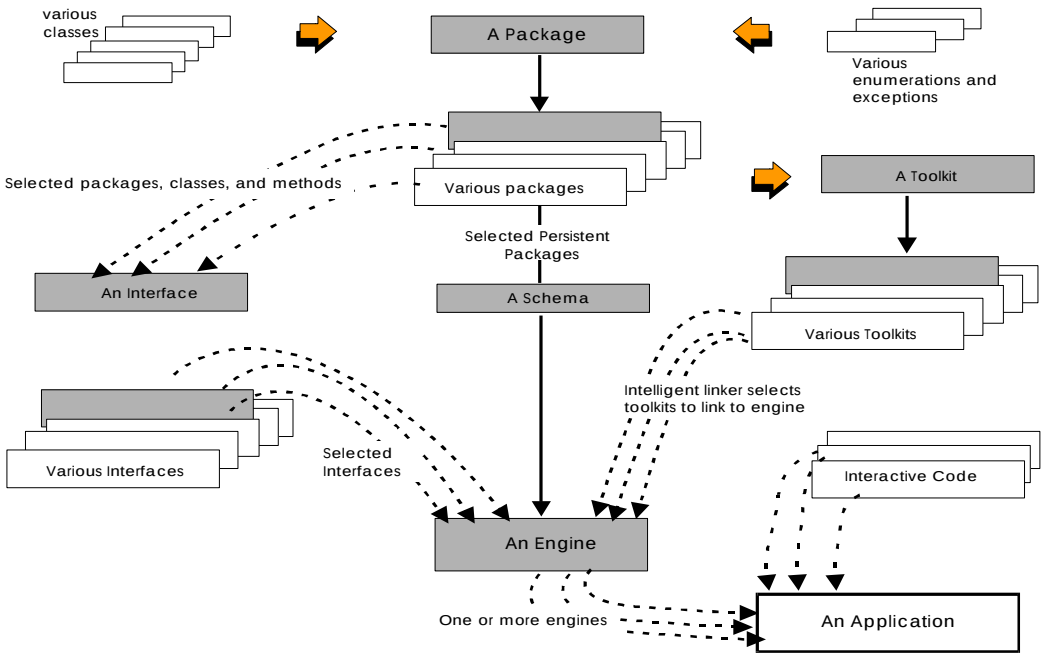


Figure 1. Development environment units and their roles.

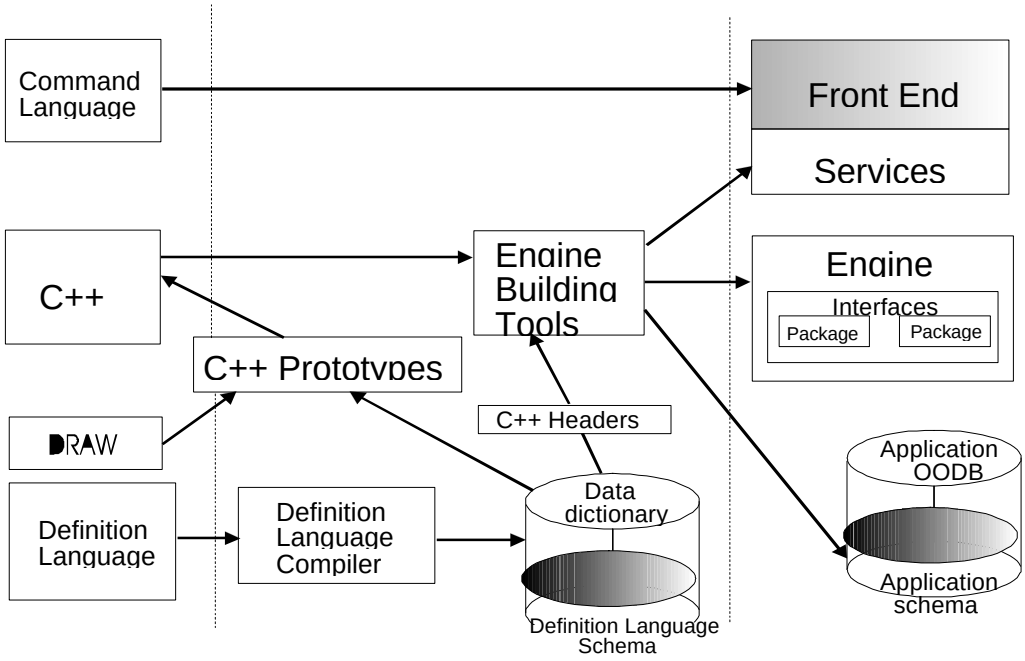


Figure 2. Development schematic

The data dictionary is central to the workings of the environment. By its presence in the dictionary, a class is available for instantiation - objects of that class can exist (C++ headers are then available). Its place in the hierarchy of classes is known, both in the software and in the database. Therefore the database structure is being built as the application is being developed.

The environment also includes a command language which is an interpretive scripting language. It is available for writing front-end scripts and for interacting directly with an application engine. It is especially useful in building the graphical user interface and graphical applications. It allows for debugging semantics and syntax without constantly re-compiling the C++ code.

One other important piece of the development toolkit is the draw environment. Draw is a wire-frame presentation environment where one can create and present objects without having the whole graphical use interface running. This allows the user to write, visualize, and debug the C++ code that executes procedures in a rapid fashion.

This toolkit provides all of the functionality required to create a stable customized CAD system with a self-consistent object-oriented data structure. Although the detailed workings of Pelorus are quite different, it too offers the same results. Fairly powerful workstations are required for these environments. For this work, the toolkit is running on a DEC AlphaStation 250 running Digital UNIX. The workstation has 2 GB of hard disk storage and 128 MB of memory.

A PROTOTYPE CAD SYSTEM FOR SHIP STRUCTURE

One of the prototypes in development is that of a preliminary design system for containerships and it is within the scope of this prototype that this discussion will proceed.

Application Specifications-

The first step in the development of a prototype is to list its specifications. These are the sequences of tasks that are used to design a piece of the product, for instance, the parallel midbody. This enumeration should be done by the current designers themselves with some input from manufacturing. This input is needed to make sure that current or proposed fabrication practices are being reflected in the design sequences and tasks. This should be done with the attitude that process improvement should always be a primary goal.

These detailed specifications should be of the intent of a task and not of the actions taken to accomplish the task. To clarify the difference, the intent of the task of creating the hullform in the parallel mid-body section of a ship could be stated as, “The hullform should have a flat portion on the bottom and a flat portion on the vertical. In between, it should have a curvature of constant sign and the whole surface should be continuous and have two continuous derivatives”. That is the detail of the task. It is the “what to do.” In contrast, a specification of action would prescribe a way of creating that surface. It could be stated by, "Create the hullform by creating a piecewise continuous polynomial surface that passes through a set of prescribed points". This is the “how to do it” and that should not be done at this initial step.

Class Hierarchies

The next step is to create the class hierarchies and appropriately assign the procedures as methods to them. Some classes are obvious while others are not and a developer needs to use the product as a guide in creating these classes.

One of the benefits of using OOP to do this development is that the standards that are in place or are developing for the exchange of ship related CAD data are very much class hierarchies. The two important ones are the application protocols for STEP and NSRP (20). The STEP AP for ship structures has not yet been adopted and it is not likely to be adopted in the near future. The NSRP standards exist and are apparently fixed. Although the basic geometric and topological entities of the development environment are based on STEP definitions, this does not include high level entities such a stiffeners, decks, bulkheads, etc. Since the STEP ship structures AP is not yet finished, the NSRP standards have been used for guidance in developing the structure class hierarchy.

As an example of the guidance found in the NSRP standard, the application object ship_edge described in section 4.2.536 is

envisioned to be a subtype of the ISO 10303 Part 42 entity. Section 4.2.566 describes a ship_seam. This is not a subtype of Part 42 but it is according to NSRP a type of ship_edge. It is clear, ship_seam is a class that inherits from the class ship_edge.

Stiffeners can be created using a profile and sweeping it along some curve to create a solid. In terms of geometry the profile is a set of curves or line segments that form a closed loop. In terms of topology these curves constitute an *edge* which bounds a *face*. The topological *face* is used to create a *solid* by sweeping or piping. This action creates new *edges* and *faces* - the defining boundary topology of a solid.

Defining a class for stiffeners allows all manifestations of stiffener to inherit the methods that can be used on this geometry and its associated topology. Classes for prismatic and curved stiffeners can be created and they will inherit from this class. In the NSRP standard, there is an application object called structural_part (4.2.691). It is the highest level object in the hierarchy of parts used to build a structure. One of the types of structural_parts is strucutral_shape_part (4.2.756). One of the types of strucutral_shape_part is structural_stiffener (4.2.785). This clearly suggests an appropriate class hierarchy.

Using NSRP application objects in this way, a nearly complete ship structures class hierarchy can be created. There need not be strict adherence but for the near future there is certainly a strong impetus to follow the NSRP standard. Even if the NSRP names are not used explicitly as class names they can be included in the class definitions as variables or attributes.

An example of one facet where adherence may not make sense is in the definitions of certain surfaces. The NSRP application objects that are used by the unit of functionality molded hullform includes hull_offset_definition, hull_surface_definition, and hull_wireframe_definition. In terms of STEP Part 42, a molded hullform is a surface which can be defined as a Bezier surface or a NURBS surface (there are other choices), but a surface cannot be defined by a set of points or by a wireframe. A set of points, a polygonal faceted surface, or a wireframe may be used to represent the hullform on a computer screen, but these are presentation methods and do not constitute a definition of a surface.

Prototype Completion

Once the hierarchy is established then the methods can be allocated to their proper places. The procedures for the tasks are now chosen and the appropriate C++ code is written. There are usually numerous ways to accomplish a task and choices need to be made with caution. The robust and efficient nature of the resultant CAD system is affected greatly by these choices. The assignment of methods demands care because of the property of inheritance. A properly placed method can help minimize the amount of code needed. As with the example regarding polygons the method for calculating and providing area could be in the class quadrilateral and the class rectangle simply inherits it and then square inherits it. The general method should be at its highest possible level in the hierarchy where as many classes as possible can inherit it. If a more specific method is desired for a subclass then it can the defined in that class.

It is at this point that the development environment is used to create the application engine in the manner described above.

A somewhat similar process is followed for the development of the hierarchy of the graphical user interface. There is much less guidance available here and the satisfaction gained from the look, feel, and functionality of an interface is very much decided by taste. The creation of user the interface is something that requires serious thought. A developer can easily create an interface that offers too many options and features and is therefore overwhelming or confusing for the user. A key philosophy in this area is “keep it simple,” - only the functionality that is truly needed should be added to the interface. “Lean and mean” interfaces are more computationally efficient and lead to more efficient use.

CONCLUSION

An enterprise-wide, rich database, of which CAD data is only a part, is the foundation of modern manufacturing methods. Information is a company’s key asset and computer-aided design systems are in the broad sense business systems which create information. They are not isolated engineering tools. To maximize company performance CAD systems need to be tailored to a company’s design, manufacturing, and business practices. CAD systems need to capitalize on a company’s strengths, help streamline and improve the design process, and shorten design cycle times. A purely customized CAD system would be best if it is possible and economically feasible.

The current choice for developing information-based applications is object-oriented technology. The power of this emerging technology lies in its features that are extremely well suited for large applications with heterogeneous data types. It is feasible for a company to develop a totally customized CAD system using commercially available object-oriented programming toolkits. These toolkits contain the needed features and tools to develop a CAD system, and without these such development would not be economically justifiable.

A shipyard can build a self-contained state-of-the-art CAD system (a smart 3D product modeling system) customized to shipbuilding and to the yard itself. There exists significant guidance on how to build the structure of such a CAD system in the standards of the NSRP and STEP.

It is feasible to do so but it is not a trivial task, even with the development environments available. It requires a clear understanding of the existing or proposed processes in the yard. It requires expertise in object-oriented programming languages and technology. Obviously having people on board who already are proficient in object-oriented programming would help a great deal, but today those people are in great demand and not easily hired or retained. It is easier for a yard’s employees to learn to program than for a yard to hire experienced programmers. Engineers and designers that can somewhat program are preferable to programmers who can somewhat engineer or design. It does not require people with 10 years of programming experience or masters degrees in computer science, but it does require training.

It very difficult to judge the time-frame of such a development or how many people it would take to build an in-house CAD system. A best guess is that 6 to 10 productive people who have been adequately trained in object-oriented programming could get a fairly sophisticated system running in 6 months.

It is not the long term goal of this research to produce a complete CAD system. Work will continue on components to clarify the feasibility of in-house development and to prove the value of object-oriented technology in design and manufacturing applications. Future efforts are planned to use the CAD database in a planning and enterprise management system, in a virtual reality environment that supports simulation based design, and in an Internet-based information interchange application. In each of these areas, object-oriented toolkits exist and each should be able to use one common database.

References

1) Marks, P. and Riley, K., *Aligning Technology for Best Business Results, A Guide for Selecting and Implementing Computer-aided Design and Manufacture Tools*, Peter Marks, Design Insight, and Kathleen Riley, Los Gatos, CA, 1995.

2) Moody, J. L., Chapman, W. L., Van Voorhees, F. D., and Bahill, A. T., *Metrics and Case Studies for Evaluating Engineering Designs*, Prentice Hall PTR, Upper Saddle River, NJ, 1997.

3) Johansson, K., "The Product Model as a Central Information Source in a Shipbuilding Environment," proceedings of the 1995 Ship Production Symposium, Society of Naval Architects and Marine Engineers, January, 1995.

4) "Development of STEP Ship Model Database and Translators for Data Exchange Between U. S. Shipyards," DARPA/MARITECH, MARITECH SOL BAA 94-44.

5) Deitz, D., "Next-Generation CAD Systems," Mechanical Engineering (magazine), vol. 118, no. 7, 1996, pp. 68-71.

6) Deitz, D., "Job Shops Retool," Mechanical Engineering (magazine), vol. 118, no. 11, 1996, pp. 78-80.

7) "Assessment of Computer Aids in Shipyards," NSRP 0373, National Shipbuilding Research Program and the Society of Naval Architects and Marine Engineers, Industrial Engineering Panel (SP-8), 1993.

8) Storch, R. L. , Clark, J., and Lamb, T., "Technology Survey of U.S. Shipyards -1994," proceedings of the 1995 Ship Production Symposia, Society of Naval Architects and Marine Engineers, January, 1995.

9) Ross, J., and Garcia, L., "The Influence of Integrated CAD/CAM Systems on Engineering for Production Methodologies in Shipbuilding," proceedings of the 1995 Ship Production Symposium, Society of Naval Architects and Marine Engineers, January, 1995.

10) Latorre, R. and Zeidner, L., "Computer-Integrated Manufacturing: A Perspective," Journal of Ship Production, vol. 10, no. 2, May 1994, pp. 99-109.

11) Storch, R. L. and Chirilo, L. D., "The Effective Use of CAD in Shipyards," Journal of Ship Production, vol. 10, no. 2, May 1994, pp. 125-132.

12) "Concurrent Engineering - Primer and User's Guide for Shipbuilding," NSRP 0435, National Shipbuilding Research Program and the Society of Naval Architects and Marine Engineers, Industrial Engineering Panel (SP-8), 1995.

13) "Concurrent Engineering - Application," NSRP 0436, National Shipbuilding Research Program and the Society of Naval Architects and Marine Engineers, Industrial Engineering Panel (SP-8), 1995.

14) "Concurrent Engineering Implementation in a Shipyard," NSRP 0454, National Shipbuilding Research Program and the Society of Naval Architects and Marine Engineers, Industrial Engineering Panel (SP-8), 1995.

15) "Evaluation of Shipbuilding CAD/CAM Systems," NSRP 4-94-1, National Shipbuilding Research Program and the Society of Naval Architects and Marine Engineers, Design Production Integration Panel (SP-4), 1994 (in progress).

16) Meyer, B., An object-oriented environment: principles and application, Prentice Hall, New York, 1994.

17) Kemper, A. H., Object-oriented database management: applications in engineering and computer-science, Prentice Hall, New York, 1994.

18) Burleson, D. K., Practical application of object-oriented techniques to relational databases, Wiley, New York, 1994.

19) Taylor, D. A., *Object-Oriented Technology: A Manager's Guide*, Addison-Wesley, New York, 1990.

20) Webster, B. F**,** *Pitfalls of Object-Oriented Development*, M & T Books, New York, 1995.

21) NSRP STEP Application Protocols, Design/Production Integraton (SP-4), March, 1996.